XGBoost Model

Audrey Garcia

2025-12-09

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.6
## v forcats   1.0.1     v stringr   1.6.0
## v ggplot2   4.0.1     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.2.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(mlr3verse)
```

```
## Loading required package: mlr3
```

```
# Save Rmd in a folder that also includes the ucla-stats-101-c-fall-2025-regression folder
amazon_purchases <- read_csv("amazon-purchases.csv")
```

```
## Rows: 1850717 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): Shipping Address State, Title, ASIN/ISBN (Product Code), Category,...
## dbl  (2): Purchase Price Per Unit, Quantity
## date (1): Order Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
fields <- read_csv("fields.csv")
```

```
## New names:
## Rows: 23 Columns: 2
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (2): ...1, fields
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
survey_train_test <- read_csv("survey_train_test.csv")
```

```
## Rows: 5027 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr (23): Survey.ResponseID, Q.demos.age, Q.demos.hispanic, Q.demos.race, Q....
## dbl  (1): Q.amazon.use.hh.size.num
## lgl  (1): test
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Problem description says:
# "removed all information from the test data that wouldn't be known at sign-up (you only have Response
# but he didn't...
# so I did here



survey <- survey_train_test |>
  select(Survey.ResponseID, Q.demos.gender, Q.demos.state, Q.amazon.use.hh.size.num, test) |>
  mutate(Q.demos.gender = as.factor(Q.demos.gender)) |>
  mutate(Q.demos.state = as.factor(Q.demos.state)) |>
  rename(Gender = Q.demos.gender, State = Q.demos.state)
```

```
amazon <- amazon_purchases |>
  rename(Survey.ResponseID = `Survey ResponseID`)
fulldata <- left_join(amazon, survey, by = "Survey.ResponseID")
```

```
fulldata <- fulldata |>
  mutate(Day = day(`Order Date`), Month = month(`Order Date`), Year = year(`Order Date`)) |>
  select(-`Order Date`, -Title, -`ASIN/ISBN (Product Code)`, -`Shipping Address State`)

pivoted <- fulldata |>
  group_by(Survey.ResponseID, Category) |>
  summarize(Count = sum(Quantity)) |>
  pivot_wider(names_from = Category, values_from = Count, values_fill = 0)
```

```
## `summarise()` has grouped output by 'Survey.ResponseID'. You can override using
## the `.groups` argument.
```

```
joined <- pivoted |>
  left_join(survey, by = "Survey.ResponseID")

training <- joined |>
  filter(test == FALSE)
testing <- joined |>
  filter(test == TRUE)
```

Model 1: Rpart Decision Tree

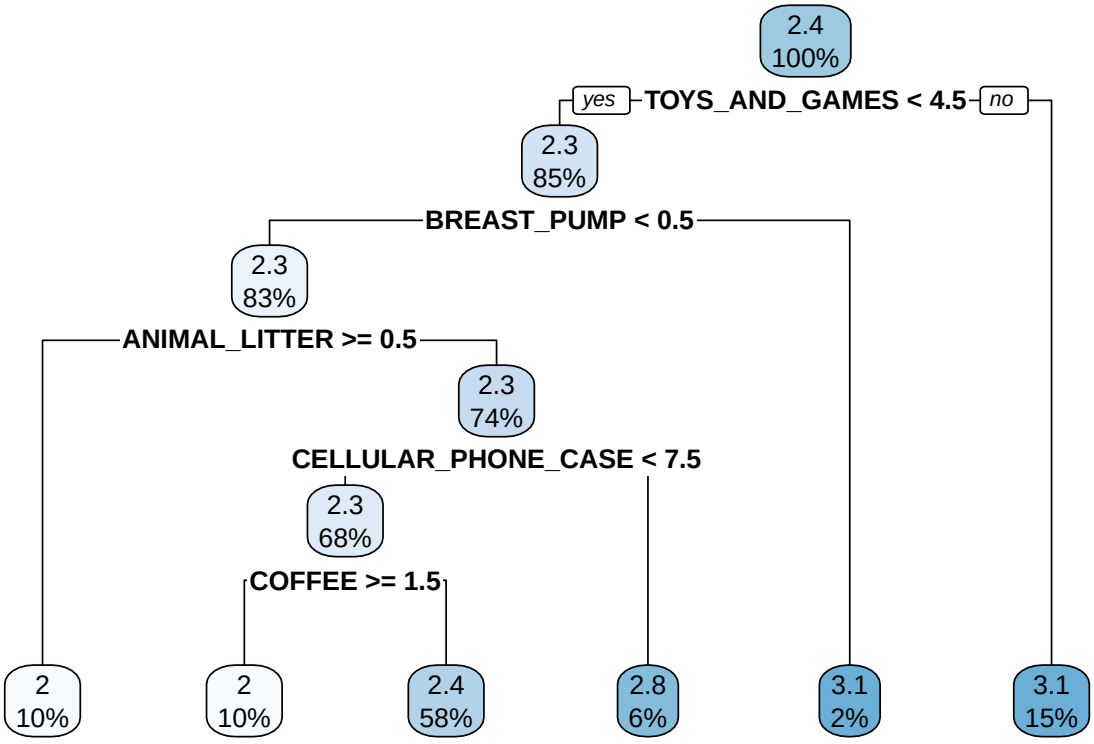
```
library(rpart)
library(rpart.plot)
rpart_task <- as_task_regr(training, target = "Q.amazon.use.hh.size.num")
set.seed(12)

tree_learner <- as_learner(
  ppl("robustify") %>>%
  lrn("regr.rpart")
)
```

```
tree_learner$train(rpart_task)

tree_learner$model$regr.rpart$model
```

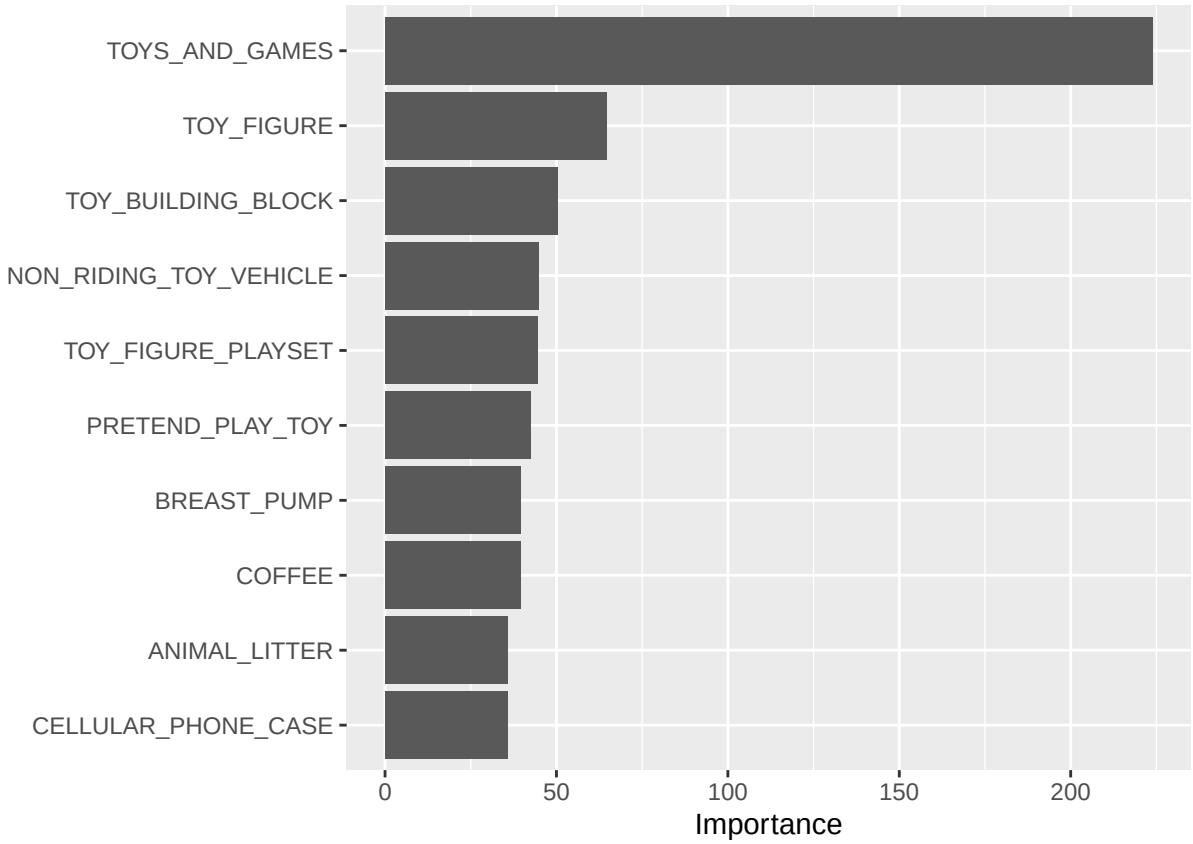
```
## n= 3027
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 3027 3659.38200 2.438718
##    2) TOYS_AND_GAMES< 4.5 2583 3005.52700 2.325978
##      4) BREAST_PUMP< 0.5 2524 2913.03400 2.307052
##        8) ANIMAL_LITTER>=0.5 289  222.83040 1.975779 *
##        9) ANIMAL_LITTER< 0.5 2235 2654.38700 2.349888
##         18) CELLULAR_PHONE_CASE< 7.5 2049 2395.72200 2.311859
##           36) COFFEE>=1.5 300  270.83670 1.976667 *
##           37) COFFEE< 1.5 1749 2085.39700 2.369354 *
##         19) CELLULAR_PHONE_CASE>=7.5 186  223.05910 2.768817 *
##      5) BREAST_PUMP>=0.5 59   52.91525 3.135593 *
##    3) TOYS_AND_GAMES>=4.5 444  430.02700 3.094595 *
```

```
tree_learner$model$regr.rpart$model |> rpart.plot()
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and is.binary
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



```
vip::vip(tree_learner$model$regr.rpart$model)
```




```
tree_learner$model$regr.rpart$param_vals
```

```
## $xval
## [1] 0
```

Model 2: random forest

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

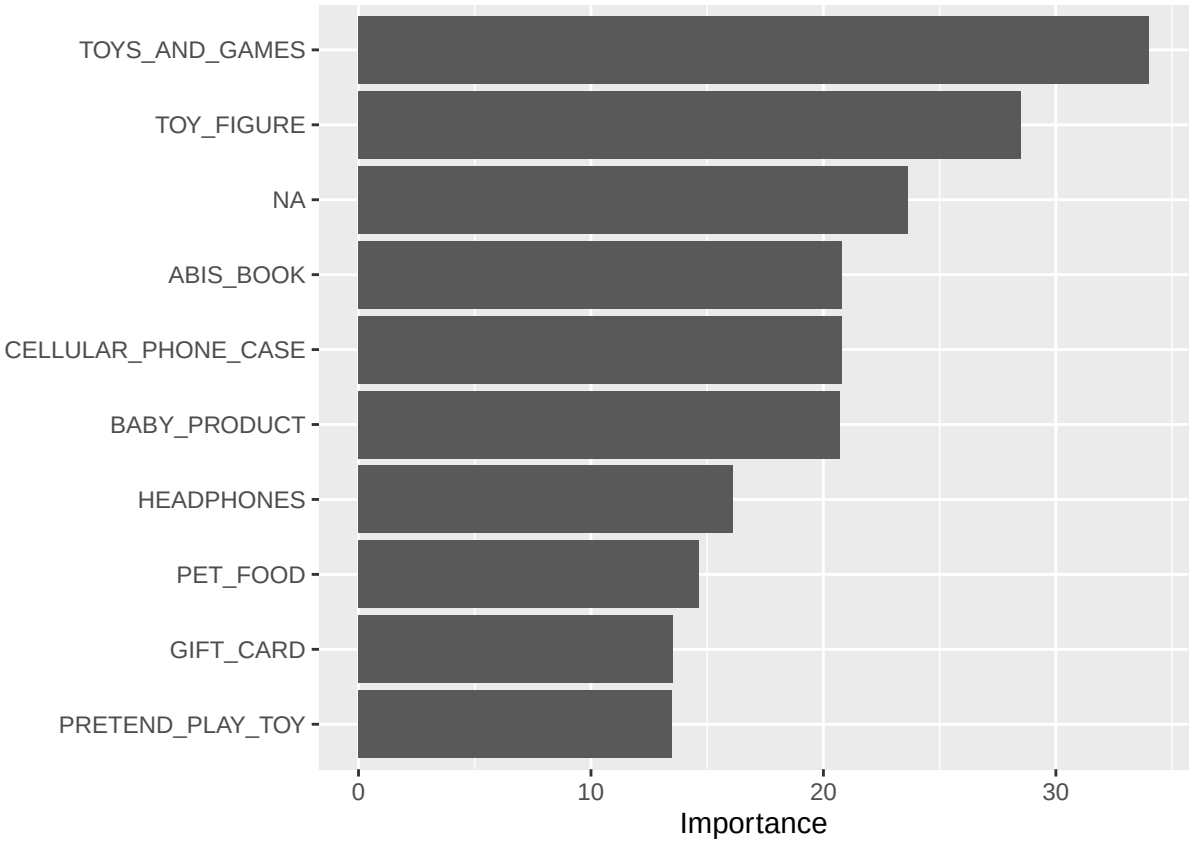
```
rf_task <- as_task_regr(training, target = "Q.amazon.use.hh.size.num")

rf_learner <- as_learner(
  ppl("robustify") %>>%
  lrn("regr.ranger", importance = "impurity")
)

rf_learner$train(rf_task)
```

```
## Growing trees.. Progress: 96%. Estimated remaining time: 1 seconds.
```

```
rf_learner$model$regr.ranger$model$model
```

```
## Ranger result
##
## Call:
##  ranger::ranger(dependent.variable.name = task$target_names, data = data,      importance = "impurity
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      3027
## Number of independent variables:  2880
## Mtry:                             53
## Target node size:                 5
## Variable importance mode:         impurity
## Splitrule:                        variance
## OOB prediction error (MSE):       1.061859
## R squared (OOB):                  0.1219324
```

```
vip::vip(rf_learner$model$regr.ranger$model$model)
```




```
rf_learner$model$regr.ranger$param_vals
```

```
## $importance
## [1] "impurity"
##
## $num.threads
## [1] 1
##
```

```
## $sigma2.threshold
## [1] 0.01
```

Final Model : XGBoost

```
xg_task <- as_task_regr(joined, target = "Q.amazon.use.hh.size.num")
set.seed(12) # for group 12 :)

lrn_xgboost <- as_learner(
  ppl("robustify") %>>%
  lrn("regr.xgboost",
    eta = to_tune(1e-4, 1, logscale = TRUE),
    max_depth = to_tune(1, 10),
    colsample_bytree = to_tune(1e-1, 1),
    colsample_bylevel = to_tune(1e-1, 1),
    lambda = to_tune(1e-3, 1e3, logscale = TRUE),
    alpha = to_tune(1e-3, 1e3, logscale = TRUE),
    subsample = to_tune(1e-1, 1)
))

at_xgb = auto_tuner(
  tuner = tnr("random_search"),
  term_evals = 700,
  learner = lrn_xgboost,
  resampling = rsmp("holdout"), ### "holdout"
  measure = msr("regr.rmse")
)
```

```
# Hyperparameter tuning
# commented out for quicker run time
# HOWEVER, to achieve exact results as submitted, proceed using the complete hypertuning process
# (roundoff errors)

# at_xgb$train(xg_task , row_ids = which(joined$test == FALSE))
# best_xgb <- at_xgb$learner$clone(deep = TRUE)
```

```
# best_xgb <- at_xgb$learner$clone(deep = TRUE)

# tuned_hyperparams <- best_xgb$model$regr.xgboost$param_vals


best_xgb <- as_learner(
  ppl("robustify") %>>%
  lrn("regr.xgboost")
)
best_xgb$pipeops_param_set_values$regr.xgboost = list(
      alpha = 0.0585235,
      colsample_bylevel = 0.3437478,
      colsample_bytree = 0.9904165,
      eta = (0.003094458),
      lambda = (0.0179533),
      max_depth = 8,
      nrounds = 1000,
```

```
      nthread = 1,
      subsample = 0.40495
)
```

```
# Cross-Validation and Error Metrics : Benchmarking
training <- joined |>
  filter(test == FALSE)
testing <- joined |>
  filter(test == TRUE)

task_cv <- as_task_regr(training, target = "Q.amazon.use.hh.size.num")

resampler <- rsmp("cv", folds = 5)

learners <- lrns(c("regr.rpart", "regr.ranger")) # finish this
learners$regr.rpart = tree_learner
learners$regr.ranger = rf_learner
learners$regr.xgboost = best_xgb

bmr_design <- benchmark_grid(tasks = task_cv,
                             learners = learners,
                             resamplings = resampler)

bmr <- benchmark(design = bmr_design)

bmr$aggregate(measures = msr("regr.rmse"))
```

```
##       nr  task_id
##    <int>   <char>
## 1:     1 training
## 2:     2 training
## 3:     3 training
##
##
## 1:   removeconstants_prerobustify.char_to_fct.POSIXct_to_dbl.ord_to_fct.imputehist.missind.impute_log
## 2:  removeconstants_prerobustify.char_to_fct.POSIXct_to_dbl.ord_to_fct.imputehist.missind.impute_logi
## 3: removeconstants_prerobustify.char_to_fct.POSIXct_to_dbl.ord_to_fct.imputehist.missind.impute_logic
##    resampling_id iters regr.rmse
##           <char> <int>     <num>
## 1:            cv     5  1.064894
## 2:            cv     5  1.028006
## 3:            cv     5  1.019687
## Hidden columns: resample_result
```

```
autoplot(bmr)
```

Training and Predicting Final Model for Submission

```
best_xgb$train(xg_task , row_ids = which(joined$test == FALSE))
```

```
prediction <- best_xgb$predict(xg_task, row_ids = which(joined$test == TRUE))
```

```
submission <- cbind(joined$Survey.ResponseID[which(joined$test == TRUE)], prediction$response) |>
  as.data.frame()
names(submission) <- c("Survey.ResponseID", "Q.amazon.use.hh.size.num")
```

```
write_csv(submission, "submission.csv")
```

```
# Plot one tree of xgb model
# commented out for pdf knitting

library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
# xgboost::xgb.plot.tree(model = best_xgb$model$regr.xgboost$model, trees = 1)
```